Týden 9

Přednáška

Připomeňme si, co to znamená rozhodnutelnost problému, v řeči „tabulek“: problém P (typu Ano/Ne) je rozhodnutelný, jestliže existuje algoritmus [Turingův stroj] M takový, že vstupně-výstupní tabulka T_M je konzistentní s tabulkou T_P (stejným vstupům odpovídají stejné výstupy).

Pro zajímavost si všimněme, že o konkrétním problému můžeme zjistit, že je rozhodnutelný, aniž jsme schopni určit konkrétní algoritmus, který jej řeší. Podívejme se např. na následující problém.

Název: 7vPi (*Sedmičky v Ludolfově čísle* π)

Vstup: Přirozené číslo k.

Otázka: Existuje v desetinném rozvoji čísla π úsek sedmiček délky k ?

Je jasné, že problému 7vPi odpovídá jedna z následujících tabulek.

Vstup	Výstup
0	Ano
1	Ano
2	Ano
3	Ano
4	Ano
5	Ano
...	...

Vstup	Výstup
0	Ano
1	Ne
2	Ne
3	Ne
4	Ne
5	Ne
...	...

Vstup	Výstup
0	Ano
1	Ano
2	Ne
3	Ne
4	Ne
5	Ne
...	...

Vstup	Výstup
0	Ano
1	Ano
2	Ano
3	Ne
4	Ne
5	Ne
...	...

...

Ke každé tabulce umíme snadno navrhnout příslušný algoritmus, takže problém 7vPi je jistě rozhodnutelný. Nevíme ovšem, která tabulka je ta pravá, takže nejsme schopni předložit konkrétní algoritmus a prokázat, že řeší problém 7vPi.

Základem k prokazování nerozhodnutelnosti problémů je krátký, byť trochu „zapeklitý“, důkaz nerozhodnutelnosti tzv. diagonálního problému zastavení:

Název: DHP (*Diagonal Halting Problem*)

Vstup: Turingův stroj M (resp. jeho přirozený kód třeba v binární abecedě).

Otázka: Zastaví se (výpočet stroje) M, když začne pracovat na (vstupním) slově, které je kódem jeho samého ?

Důkaz nerozhodnutelnosti DHP (sporem). Předpokládejme, že tento problém je rozhodován Turingovým strojem D. Pak můžeme využít D jako proceduru a sestrojit Turingův stroj X, který se chová takto: pro zadané slovo w nejdříve zkontroluje, zda w je kódem Turingova stroje, a když ano, tak pomocí D zjistí, zda onen zadaný stroj se na svůj kód zastaví

či nezastaví; v případě kladné odpovědi od procedury D se X nezastaví (provádí např. nekonečný cyklus), v případě záporné odpovědi se zastaví. Zkusme teď zodpovědět otázku, zda se X zastaví, když je mu na vstupu předložen jeho vlastní kód. Jelikož nutně dojdeme k logickému sporu, stroj D nemůže existovat.

Převeditelnost mezi problémy

Ujasnili jsme si (i využitím tabulek problémů), co to znamená, když se řekne, že

> problém P_1 je *algoritmicky převeditelný* (stručněji *převeditelný*) na problém P_2; označujeme $P_1 \rightsquigarrow P_2$: existuje algoritmus, který k instanci I_1 problému P_1 sestrojí instanci I_2 problému P_2 tak, že odpověď na otázku pro I_1 v P_1 je stejná jakou odpověď na otázku pro I_2 v P_2.

Ilustrovali jsme si na případu DHP$\rightsquigarrow$HP, kde HP je definován níže. Vyvodili jsme, že HP je také nerozhodnutelný (využitím tvrzení 6.11. v části 6.5.).

NÁZEV: HP (*Halting Problem*)

VSTUP: Turingův stroj M a (vstupní) slovo w.

OTÁZKA: Zastaví se M na w ? (Je tedy výpočet M pro vstup w konečný ?)

Poznámka. Obecně je přirozené definovat, že P_1 je převeditelný na P_2, jestliže existuje algoritmus rozhodující P_1, pokud může využívat (hypotetickou) proceduru rozhodující P_2; to je tzv. *turingovská převeditelnost*. Často ale stačí speciální případ, který je uveden výše; ten bereme jako základní.

Částečná rozhodnutelnost, Postova věta

Připomněli jsme *částečnou rozhodnutelnost* problémů. Přitom byla podstatná následující definice, kterou zde formulujeme v řeči „tabulek“ (problému P odpovídá tabulka T_P, stroj M přirozeně definuje tabulku T_M):

> Turingův stroj M *částečně rozhoduje* problém P (typu ANO/NE), jestliže u každého vstupu, pro nějž je v T_P ANO, je v tabulce T_M výstup ANO a pro každý vstup, pro nějž je v T_P NE, je v T_M výstup NE nebo znak $\perp$ (nedefinováno).

(Pro vstupy, kterým problém P přiřazuje odpověď NE, nemusí stroj M svůj výpočet skončit.)

Jak se dá očekávat, o problému řekneme. že je *částečně rozhodnutelný*, jestliže existuje algoritmus (Turingův stroj), který jej částečně rozhoduje.

Speciálně jsme si uvědomili Postovu větu (Věta 7.2.)

Uvědomili jsme si také, že problém HP je částečně rozhodnutelný (viz Univerzální TS), a že tedy $\overline{\text{HP}}$ (doplňkový problém k problému HP) není (ani) částečně rozhodnutelný.

Univerzální Turingův stroj

Algoritmus, kterým jsme prokazovali částečnou rozhodnutelnost problému zastavení (HP) byl tento: na zadaný stroj (program) M a vstup w spusť „interpret“, který provádí činnost (výpočet) M na vstupu w. Takový interpret je ovšem také program, tedy algoritmus, a šlo by jej proto realizovat (naprogramovat) ve formě konkrétního Turingova stroje U (viz Věta 7.3.).

Riceova věta

Uvědomili jsme si, že každá vlastnost V Turingových strojů (např. „stroj M má více než 100 stavů“, „výpočet stroje M je pro každý vstup konečný“, apod.) rozdělí množinu všech Turingových strojů na dvě disjunktní podmnožiny: jedna je množina strojů, které vlastnost V mají, a druhá je množina strojů, které vlastnost V nemají. Vlastnost V je *triviální*, jestliže je jedna z oněch dvou příslušných množin prázdná (tedy buď všechny stroje vlastnost V mají nebo ji nemá ani jeden). Vlastnost V je *netriviální*, jestliže ji alespoň jeden stroj má a alespoň jeden stroj nemá.

Důkladně jsme si promysleli, co to je *vstupně/výstupní vlastnost*, zkráceně též *I/O vlastnost*, Turingových strojů (či obecně „programů“).

Speciálně jsme si uvědomili, že vlastnost V není I/O vlastností právě tehdy, když existují dva stroje M_1, M_2, které mají stejnou I/O tabulku (tedy stejné vstupně/výstupní chování), ale jeden z nich vlastnost V má a druhý ji nemá.

Probrali jsme si pak (níže uvedené) vlastnosti v řešeném příkladu 7.1. a uvědomili si, které jsou I/O vlastnostmi. Speciálně jsme si všimli, že podle definice je každá triviální vlastnost I/O vlastností.

a/ Zastaví se M na řetězec 001 ?

b/ Má M více než sto stavů ?

c/ Má v nějakém případě výpočet stroje M více kroků než tisícinásobek délky vstupu ?

d/ Platí, že pro libovolné n se M na vstupech délky nejvýše n vícekrát zastaví než nezastaví ?

e/ Zastaví se M na každém vstupu w za méně než $|w|^2$ kroků?

f/ Je pravda, že pro lib. vstupní slovo M realizuje jeho zdvojení ?

g/ Je pravda, že M má nejvýše sto stavů nebo více než sto stavů ?

Pak jsme se zamysleli nad Riceovou větou:

Každá netriviální vstupně/výstupní vlastnost programů je nerozhodnutelná.

Partie textu k prostudování

Univerzální Turingův stroj, Riceova věta (v části 7.).

Cvičení

Příklad 9.1

Vysvětlete, co to je doplňkový problém k problému P (typu ANO/NE). Pak konkrétně definujte doplňkový problém Non-Eq-CFG k problému Eq-CFG (ekvivalence bezkontextových gramatik).

Příklad 9.2

Vysvětlete podrobně, co to znamená, když řekneme, že HP$\rightsquigarrow$Non-Eq-CFG (tj. že Halting Problem je převeditelný na problém Non-Eq-CFG). (Převeditelnost HP$\rightsquigarrow$Non-Eq-CFG zde nedokazujeme, ale dále ji bereme jako fakt.)

Příklad 9.3

Je možné, že oba problémy Eq-CFG a Non-Eq-CFG jsou částečně rozhodnutelné? Umíte prokázat částečnou rozhodnutelnost alespoň jednoho z nich?

Příklad 9.4

Vysvětlete, co dělá univerzální Turingův stroj U, když dostane jako vstup slovo tvaru uv, kde slovo u je kódem stroje U.

Příklad 9.5

Uveďte alespoň tři vlastnosti Turingových strojů, pro něž plyne nerozhodnutelnost z Riceovy věty, a alespoň tři vlastnosti, pro něž nerozhodnutelnost z Riceovy věty neplyne. (Jiné příklady než byly na přednášce.)

Příklad 9.6

(Nepovinně.)
Navrhněte Turingův stroj M s jednostranně nekonečnou páskou, který pro dané vstupní slovo $w \in \{a,b\}^*$ sestrojí (výstupní slovo) $w(w)^R$. Stroj M tedy realizuje příslušné (vstupně/výstupní) zobrazení $f_M : \{a,b\}^* \to \{a,b\}^*$ (např. $f_M(abb) = abbbba$).
Pak zvolte vhodné kódování slov v abecedě $\{a,b\}$ tak, aby analogické vstupně/výstupní zobrazení mohl realizovat RAM. Navrhněte konkrétní RAM M', který toto zobrazení realizuje; přitom postupujte tak, že RAM M' přímočaře simuluje Turingův stroj M. (Nejde o co nejjednodušší RAM pro daný úkol, ale o to, abyste aplikovali obecný postup prokazující, že každý TS je možné simulovat RAMem.)